

Q31 : 下一个排列

1. 654321 : 是最大排列
2. 146532 : 是 14 开头的最大排列, 此时递增只能用6532中大于4的最小元素5开始 : 15开头
3. 而15开头的最小字典序为 15 2346 . 即交换大于4的最小元素5与4的位置, 然后后边构成递减序列, 逆置该递减序列, 即为最小字典序。

39 : 组合总和 无重复数组, 数字可以重复选取, 拼target, 结果不可重复。

0. 两种写法, 对应两种递归树结构。2,3,6,7

1. 对 cur 在cur层, 直接对cur选多次(包括0次), 第一层就是分为 对第一个数, 选择次数的不同, 归为不同分支。第二层就是在选择第一个数的基础上, 选择第二个数次数的不同, 分为不同分支。按照cur的不同选择次数划分分支。

```

      root
    ' '      2      22      222      选2
  
```

" 3 33 23 223 选3 " 3 选6 "7 选7

2. begin写法。规则是, 第一层是从 0到end选不同的数, 分不同的枝。

```

      root
    2   3   6   7   root从0开始选
  22  23   3   6
 223  23
  
```

每个节点向下一个结点发展时, 他上次选择的最后一个位置就是下次选择的起始位置。

3. 这题两种写法都好实现, 且复杂度一样、

Q40 : 组合总和 II : 有重复元素数组, 每个数字只能选一次, 拼target, 结果不可重复。

1. 每个元素分为选或者不选进行分支。一层只指针cur一个元素, 分两支。如果有两个2. 选第一个不选第二个, 和不选第一个选第二个 形成了重复。这样就很难去重。选和不选其实就是上边的 选0次和选多次, 不过这题规定最多一次。

改进: 避免重复: 如果我们按顺序对每一个元素, 使用选或者不选的策略, 那么一定会有重复。

改变策略, 我们将 相同的数, 改为, 选0次, 选1次, 选多次, 可以先使用一个map统计 记录其最后一个的下标位置。

```

int = cur;
for(; i <= map.get(num[cur]); i++){
    dfs(map.get(num[cur])+1, sum += num[cur]*(i-cur+1))
  
```

```
}

```

****限制：** 父子结点不能选择相同的数字。 **

另外：若每个数只有一个，不重复。 那么这个树，实质上是一颗二叉树。正因为有k各。
那么每个结点可以分出k+1个分支，每个分支选中 i+1个。
该问题其实就是Q39的第一

2. begin写法 + 同节点的子节点不用同元素， 每个节点向下一个结点发展时， 他上次选择的最后一个位置就是下次选择的起始位置。 当前层按照起始位置向后每一个可选元素进行分支。

方法2以然会产生重复， 有两个 2 选第一个不选第二个， 和不选第一个选第二个形成了重复。

345556

对于一颗不限制选择的递归树

```

          root
        3    4    5        5    5    6

```

可以发现当前层在选择时， 由于可以选择同结点同层已经选择过的数字
34 35 36 ... 55 55 56 55 56 56
那么3个5后边的分支必然会存在重复。

****限制同父的结点， 已经用的数字不能再用**** 。 但上层用过的数字不影响下层使用。

```

          3        4        5        6
        34 35 36, 45 46, 55 56

```

只有 同一个结点的子节点不能使用相同的元素。
这样就防止了重复。这样就可以保证 55 只在一处出现。

Q216：组合总和 III： 找出所有相加之和为 n 的 k 个数的组合。组合中只允许含有 1 - 9 的正整数，并且每种组合中不存在重复的数字

直接用：加入cur,不加入cur,分两枝条递归即可。

Q46：重复元素的全排列： 不需要考虑去重。

不同于上边的Q39, Q40， 本题是每层选一个， 本分支祖先没有使用过的数字加入。所以直接使用一个vis数组做标记即可。 选一个， 递归， 返回， 回溯， 再选一个。这样在一层做出不同的选择， 对应了不同的排列。 因为无重复元素， 不需要考虑去重。

Q51：N 皇后

R[N]：对行占位 C[N]:对列占位 LL[2*n-1]:占主对角线 RR[2*N-1]: 占副对角线。
进行回溯， 当前层要占用一个格子， 必须满足， 以上四个条件未占用。

我们通过回溯， 在每一行选一个可行的格子， 且使其满足列和对角线3个占位条件即可。 因为按行占位， 行一定不会冲突。

Q60 : 排列序列 : $1 \sim n$ 的全排列, 按照字典序, **输出第k个排列**, 从第一个开始。又名: 第k个字典序排列

hard 从高位开始确定, 有一个很直观的点就是, k 越小, 高位越小。我们确定第一位数。如果 $k \leq (n-1)!$: 那么首个元素一定是1, 因为后边还有 $n-1$ 个数的全排列, 恰好有 $(n-1)!$ 种。如果 $(n-1)! < k \leq 2*(n-1)!$ 那么首元素一定是2, 因为首元素1有 $(n-1)!$ 种, 2开头的也有 $(n-1)!$ 种。依次类推。然后我们确定第二位。假设我们确定了第一位是3, 那么1到 n 种数字3就被使用过了。而且以1,2开头的总共有 $2*(n-1)!$ 种。我们要找的范围就是以3开头的 $(n-1)!$ 种的其中之一。所以我们直接对 $(n-1)!$ 求余数, 直接在第3个 $(n-1)!$ 的范围内找。于是我们只需要对 $k = k \% (n-1)!$ 然后在首位是3的情况下, 找后序的第 k 排列即可。

如果 $5*(n-1)! < k \leq 6*(n-1)!$ 那么就是找第5小 (因为3用过了, 不能直接用5. 第五小此时是6.)

Q77 : 组合 返回 $1 \dots n$ 中所有可能的 k 个数的组合

1. 增量构造法 : 比较难理解。其实就是begin写法。即一个结点的子节点, 选取元素时可能上层最后一个元素之后的任意一个元素。
 2. 位向量法 : i 是或者否 存在于集合中。
 3. 二进制法 : 这个就不说了。位数必须小于32
- 这题用位向量法就挺好, 方便控制集合中元素个数。

Q78 : 子集 : 无重复元素数组, 求所有子集。

用二进制法最好的。

Q79 : 单词搜索 : 在字符矩阵中, 找单词。

其实就是对矩阵进行 dfs 的回溯搜索。

Q90 : 子集 II : 含重复元素数组, 求所有子集, 子集不可重复。

标准的begin写法, 避免重复。限制: 同父的子节点, 不能重复使用同一个数字。

Q131 : 分割回文串 : 对串分割, 要求所有子串都是回文串, 返回所有方案。

1. 首先, 我们通过dp获得所有子串是否是一个回文串。 $dp[i][j] = true$ 则子串 ij 为回文串。

2. 通过回溯的 begin写法，以当前位置为起点，准备分割下一个回文串。终点有很多选择。只要 $dp[i][j] = true$ 即可。
3. 当 $cur == len$ 时，说明已经分割完毕。存入list即可。

Q140 : 单词拆分 II

单词拆分II

1. 这个题不是求是否可以划分，而是得出所有的划分结果，所以动态规划就不行了，要使用回溯法递归了。但是可以参考
2. 回文串的划分，先使用动态规划得到可划分区域 $dp[i]$ 在 $dp[i]$ 可划分的位置进行切割。其余位置直接跳过。

Q437 : 路径总和 III : 在二叉树中，找从上到下，路径和为 sum的所有路径总数。

前序

1. 使用一个map记录当前路径一直到root结点的前缀和。使用回溯保证，map中只存当前结点到root的前缀和。
2. 这个map的作用，只保留，当前结点的祖先结点的前缀和。使用回溯保证。
3. 前序到达一个结点后，在map中找 $currSum(当前结点头前缀和) - target$ 。map中val代表所找父节点的前缀和为key的数量。

Q679 : 24 点游戏 4个数字(1到9)，你需要判断是否能通过 $*$, $/$, $+$, $-$, $(,)$ 的运算得到 24

0. 数字不可重复用，必须用4个数字。符号可以重复用。
1. 选出两个数，做运算。把结果放回，剩余三个数。
2. 选出两个数，做运算。把结果放回，剩余两个数。
3. 选出两个数，做运算。把结果放回，剩余一个数。判断该数是不是 24.

所以可以发现这是个递归的过程。每次递归，都要用双层for循环选出两个数。两个数之间有4种运算。(减法和除法，还有被减，被除)分别对6个结果，再进行下一次递归。

4. 如果等于24，返回true. 每层有4到6个递归。每个递归都返回 true,或者false. 有一个返回true. 就直接返回 true.

Hj89 : 24点游戏升级版本：扑克牌：2-10, J(11)Q(12)K(13) A(1) 共13个数。只使用 $*/+-$ 四个符号是否存在等式 得到 24.

0. 没有括号，4个符号运算级别相同，意为只能从左到右做运算。

1. 该问题只能通过 暴力枚举的方式来做。枚举4位数字的顺序，然后枚举3个符号的总类。然后从左到右进行计算。
2. 4个数字，4层for,且要求枚举时不能重复使用元素。4个数字全排列。
3. 3个符号，3层for,符号可以重复。就是可重复选，选出3个。总共有 444总可能
4. 所以用for暴力枚举即可。
5. 数字为什么也要改变顺序：因为计算顺序只能从左到右，那么顺序不同时，可能只有一种排列能找到3个符号得到24，另一种排列可能就是none.